Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: 5970490.pn.

Display: 50 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Tuesday, September 04, 2007 **Purge Queries** Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L14	5970490.pn.	2	L14
L13	L12 and sibling$	7	L13
L12	L11 and (descendant$)	14	L12
L11	L10 and (child near attribute)	49	L11
L10	L9 and (attribute near type)	588	L10
L9	l1 and ("tree structure")	5340	L9
L8	L3 and (descendant$)	5	L8
L7	L5 and ancestor$	0	L7
L6	L5 and sibling$	0	L6
L5	L4 and (descendant$)	4	L5
L4	L3 and (attribute near type)	13	L4
L3	L2 and (child near attribute)	16	L3
L2	L1 and (intellectual near property)	3479	L2
L1	707/$.ccls.	60796	L1

END OF SEARCH HISTORY

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L5 and (child near attribute)

Display: 100 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Monday, September 03, 2007 **Purge Queries** Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L6	L5 and (child near attribute)	17	L6
L5	L4 and sibling	565	L5
L4	L3 and (not descendants)	25082	L4
L3	707/$.ccls.	25855	L3
L2	L1 and option	1	L2
L1	6845383.pn.	1	L1

END OF SEARCH HISTORY

9/28/01

L14: Entry 1 of 5 File: USPT Jun 12, 2007

DOCUMENT-IDENTIFIER: US 7231394 B2
TITLE: Incremental bottom-up construction of data documents

PRIOR-PUBLICATION:

DOC-ID	DATE
US 20030028557 A1	February 6, 2003

Brief Summary Text (13):
A content description may be updated by adding, deleting or replacing description fragments, i.e., descriptors and description schemes, and/or attributes within fragments. The updates are transmitted to the receiving system through a series of packets, or "access units" in the MPEG-7 standard, that contain one or more fragment update units. The decoder on the receiving system updates its existing content description by applying the information in the fragment update units. Typically a fragment update unit consists of a navigation path that directs the decoder to the appropriate locations in the description tree to apply the update, an update command that specifies the type of update to execute, i.e., add, delete, replace, and a fragment payload that identifies the update value for an add or replace command. Because every current update command must specify the correct path to the update locations, the encoder must have prior knowledge of the description tree stored in the decoder before creating and transmitting the fragment update units. Thus, the current fragment update units can only construct the description tree at the decoder from the top down.

Description Paragraph (9):
A data document having a tree structure may be constructed from the bottom-up by merging and/or joining nodes in an existing data document with nodes in a document fragment as described herein. Examples of such documents include XML (Extensible Markup Language) documents and multimedia content description documents, in particular those containing descriptions complying with the MPEG-7 standard. Exemplary embodiments are described in terms of multimedia content description documents but the invention is not so limited and one of skill in the art will immediately recognize the applicability of the invention to other types of data documents.

Description Paragraph (10):
Beginning with an overview of a multimedia content description delivery system 100 as illustrated in FIG. 1A, a content description 113 on a transmitting system 101 is re-created on a receiving system 105 using description fragments transmitted as access units 109. The content description 113 on the receiving system is constructed by a decoder 107 from the bottom up by merging and/or joining fragments in the access units 109 with an existing content description 111 on the receiving system 105 as described further below. An encoder 103 on the transmitting system creates the access units 109 containing the merge and/or join commands and the fragments as described next in conjunction with FIG. 1B. It will be appreciated that although "access unit" is the term used by the MPEG-7 standard, the invention is not so limited and is applicable to the construction of any type of content description. It will be further appreciated that the invention is not limited to the particular arrangement of components illustrated in FIG. 1A. For example, one

of skill in the art will immediately recognize that the transmitting system may receive the access units from another system for subsequent transmission

Description Paragraph (13):
As illustrated in FIG. 1D, a join command joins a fragment in the fragment update unit with a node in an existing description tree to produce a joined description tree. Given an existing description tree 151 and fragment 153 in the fragment update unit, assume that node B in the existing description tree 151 and the fragment 153 are to be joined into a single join node A' in a joined description tree 155. The navigation path 127 contains a context expression that specifies the node B in the description tree 151 as the update location at which to join the fragment description 153. After the fragment description 153 is joined to node B, the result is join node A' in the joined description tree 155. Because each of the nodes A and B may have attributes and sub-trees as children, the join source element 137 in the fragment update unit 125 specifies which children from the two nodes being joined are to be attached as child nodes for the join node A', and in what order they will appear. When a child node is added to the join node, the complete subtree rooted at the designated child is added, i.e., adding a child node includes adding all descendant nodes of the designated child node. In the simplest case, the join operation may result in the concatenation of the children of the two nodes as the children of the join node A' in the following order: first, all children of designated join node in description 153 (i.e. node B), followed by the child nodes of node A. The ordered list of source, element pairs 139 in the join source element 137 enables arbitrarily complicated joins. The source element of a pair specifies that either the existing tree 151 or fragment 153 is the source of a child of the join node A', while the join path element specifies which attributes or sub-trees of the source are to become children of the join node. Thus, as illustrated in FIG. 1D, the join source list contains two source, child path element pairs that specify how to combine the children of node B in the existing description tree 151 and node A in the fragment 153. The first pair indicates that the source is the existing description tree 151 and the child path points to child node D (specified relative to its parent node B) in tree 151. The second pair indicates that the source is the fragment and the child path points to child node C of node A in fragment 153. The resulting child nodes of the join node A' are the subtree rooted at node D, which includes nodes F and G from the existing description tree 151 and the subtree rooted at nodes C, which includes node H from the fragment 153. Note that node E from the tree 151 is not included in this case because it was not designated in the join source list.

Description Paragraph (15):
The following description of FIG. 2 is intended to provide an overview of computer hardware and other operating components suitable for implementing the invention, but is not intended to limit the applicable environments. FIG. 2 illustrates one embodiment of a computer system suitable for use as the transmitting and/or receiving system of FIG. 1A. The computer system 40 includes a processor 50, memory 55 and input/output capability 60 coupled to a system bus 65. The memory 55 is configured to store instructions which, when executed by the processor 50, perform the methods described herein. The memory 55 may also store the access units. Input/output 60 provides for the delivery and receipt of the access units. Input/output 60 also encompasses various types of computer-readable media, including any type of storage device that is accessible by the processor 50. One of skill in the art will immediately recognize that the term "computer-readable medium/media" further encompasses a carrier wave that encodes a data signal. It will also be appreciated that the system 40 is controlled by operating system software executing in memory 55. Input/output and related media 60 store the computer-executable instructions for the operating system and methods of the present invention as well as the access units. The encoder 103 and decoder 107 shown in FIG. 1A may be separate components coupled to the processor 50, or may be embodied in computer-executable instructions executed by the processor 50. In one embodiment, the computer system 40 may be part of, or coupled to, an ISP (Internet

Service Provider) through input/output 60 to transmit or receive the access units over the Internet. It is readily apparent that the present invention is not limited to Internet access and Internet web-based sites; directly coupled and private networks are also contemplated.

Description Paragraph (20):
The join method 330 illustrated in FIG. 3C replaces the node (and any of its subtrees) designated in the navigation path with a join node (E) (block 331). The element and schema type for the join node are that of the root node of the fragment. For each pair in the ordered list in the join source element, the join method 330 executes a pair loop beginning at block 333 and ending at block 345. Within the pair loop, the source element is used to determine the current source, either the existing description tree or the fragment, and each child node for the current source is evaluated in a child loop starting at block 335 and ending at block 341. If the child node matches the join path element at block 337, the child node is added to a list Q at block 339. When all the child nodes of the current source have been evaluated, the child nodes in list Q are appended to the join node E, with each being appended as the last child (block 343). If the same child attribute occurs more than once during the joining process, the first child attribute is appended and subsequent duplicate child attributes are ignored. When the last pair has been evaluated, all child nodes for the join node have been appended and the join process is complete.

Description Paragraph (24):
When the content descriptions are coded in XML, such as MPEG-7 content descriptions, in one embodiment the navigation and join paths are location path expressions in the XML path language (XPath). An XPath location path consists of a set of location steps that walk through nodes in a tree structure. Each step is relative to a set of context nodes specified by the previous step. A location step consists of three parts: 1. an axis that specifies the relationship in the tree between the nodes selected by the step and the context node(s), e.g., parent, child, ancestor, sibling, attribute, etc.; 2. a node test that specifies the node type of the nodes selected by the step, e.g., text, attribute, element, etc.; and 3. zero or more predicates that are used to filter the nodes selected by the step, e.g., * (select all), text, @attribute-name, order-number, etc. In another embodiment specific to the BiM encoding of MPEG-7 access units, the paths are relative to the XML Schema type corresponding to the type of the fragment.

Description Paragraph (26):
TABLE-US-00001 <complexType name="FragmentUpdateUnitType"> <element name="Navigation" type="mpeg7:XpathType" minOccurs="1"/> <element name="JoinSource" minOccurs="0" maxOccurs="unbounded"> <simpleContent> <simpleType> <extension base="mpeg7:XpathType"/> <attribute name="source"> <simpleType> <restriction base="string"> <enumeration value="current"/> <enumeration value="fragment"/> </restriction> </simpleType> </attribute> </simpleType> </simpleContent> </element> <element name="UpdateCommand" type="mpeg7:UpdateCommandType" minOccurs="1"/> <element name="FragmentPayload" type="mpeg7:FragmentPayloadType" <minOccurs="0" maxOccurs="1"/> </complexType>

Current US Original Classification (1):
707/101

Current US Cross Reference Classification (1):
707/1

Current US Cross Reference Classification (2):
707/E17.005

Current US Cross Reference Classification (3):
707/E17.011

Current US Cross Reference Classification (4):
707/E17.028

Previous Doc Next Doc Go to Doc#

Col. 9, l. 60-67
Col. 12, (l. 65-64
Col. 43, (l. 20)

L13: Entry 7 of 7 File: USPT Oct 19, 1999

DOCUMENT-IDENTIFIER: US 5970490 A
TITLE: Integration platform for heterogeneous databases

Detailed Description Text (92):
Thus for example, a relational data model would have an interpretation in an LSD where I consists of nodes which denote relation names, nodes which designate attribute names within the relations, and edges would connect relation nodes and attribute nodes. Similarly, an object model would designate nodes as object types, attributes, or methods, and edges would designate either: 1) a named relationship between an object type and one of its attributes, 2) a subtype (specialization) relationship between an object type and its parent object type, or 3) an aggregation relationship between an object type and its component object types.

Detailed Description Text (141):
For the Rewrite operation, if ApplyFcn returns False the visited node is not copied; if it returns True, then the current node is copied as is. If ApplyFcn returns another value, then it is taken as the value to which this node has been rewritten, and thus replaces this visited node in the output of this Rewrite operator. In general, the result for the Rewrite operator is a tree structure which is a projection of the structure of the original tree (i.e., some nodes may be omitted). The values for nodes of the rewritten tree may be transformed values or copies of the corresponding node of the original tree.

Detailed Description Text (184):
The dependency graph 200 shown in FIG. 4 is the basis for rule execution. It is implemented as a high level abstract class for the information bridge. It is composed of three submodules for the input tree 210, rule graph 220, and output tree 230, respectively. Although the term tree is used, the input and output actually may be cyclic directed graphs, though they most often are trees. Non-tree structures arise when two or more leaf nodes are identical (corresponding to multiple parents) or when there is a directed cycle representing a recursive structure, such as the contains or part-of relationships that arises in applications such as bill of materials. These three submodules are linked together to form the actual dependency graph data structure and associated functions, as shown in FIG. 4. The input tree feeds into the rule graph which feeds into the output tree.

Detailed Description Text (186):
When the accessor functions retrieve data instances, they build an instance tree which has the same logical structure as the schema tree, but now with potentially multiple instances nodes for those schema nodes which are subordinate to an aggregation node in the tree. There would be one input instance aggregate node for each occurrence of a collection--say each course--and one data node for each member of an aggregate (e.g. each student). Similarly, to represent a relation in a relational database, there would be a separate schema node for the relation itself, for a generic tuple, and for each attribute type. In the instance tree there would be as many tuple nodes as there are tuples, and each tuple would have one instance node for the value of each attribute of the relation

Detailed Description Text (214):
The schema tree represents the logical structure of the data. The structure of the schema tree, specifically the parent/child/sibling relations, are represented by the productions in the HLDSS file. The right-hand side elements of a production are children of the left-hand side. And that left-hand side, in turn, is a child of the production in which it appears as a right-hand side. All elements that appear together on the right-hand side of a production are siblings.

Detailed Description Text (239):
The data access and parsing process begins with the top or maximal node of the Logical Structure Diagram--if the LSD is a tree, this is the natural root; if the LSD is a graph, then this is a designated node such that all other nodes of the LSD are reachable as descendants of this node.

Detailed Description Text (243):
Thus the execution of the parser controller nodes provides the postorder traversal of the tree down to the level of uniform regions. Then the associated parser for the particular type of uniform region is invoked. It traverses the nodes of the uniform region, and constructs a corresponding instance subtree/subgraph. The uniform region parsers typically are specialized, and interpret the LSD schema tree relative to the specific annotations of that region. The nodes and edges may be treated differently in different regions, and even different edges and nodes within a region may have different meanings based upon the annotations. In contrast, each parser controller node treats its immediate descendants uniformly, and this applies recursively downward until a uniform region or terminal node is encountered.

Detailed Description Text (279):
The Browser example in FIG. 5 shows that the data values at the leaves of the schema are displayed. Data is associated only with the leaves, while the tree structure represents relationships in the source and/or target databases. Notice that only one set of related data values is shown at a time--that is, one data value for each leaf node in the LSD schema tree.

Detailed Description Text (312):
Of course certain semantic ATTributes, such as type, will have meaning under just one INTERPretation or the other. Thus the INTERPretation attribute can be used by a reasoning or processing engine to determine how the other attributes and constructs of the specification should be interpreted and utilized.

Detailed Description Text (358):
The main structural representation focuses on hierarchical tree-structures as the dominant structuring mechanism. It also accommodates full directed graph structures through cross-referencing among nodes in the spanning tree of the graph.

Detailed Description Text (431):
The `Local Attribute` is this rightmost attribute term in an attribute path. The `Parent AttributePath` is formed by removing the LocalAttribute from the AttributePath`. For top level elements, the Parent is shown as null, which means that the Frame is the parent. Several of the entries have been expressed separately for indexing purposes--so that one can easily find all components of a Frame, all children of an attribute path, the parent of a path, and all local attribute names.

Current US Original Classification (1):
707/10

Current US Cross Reference Classification (1):
707/104.1